

08-28-00

08/25/00
JC583 U.S. PTO

JC586 U.S. PTO
09/649437
08/25/00

Please type a plus sign (+) inside this box [+]

PTO/SB/05 (12/97)

Approved for use through 09/30/00. OMB 0651-0032

Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL
(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No. 003242.P017 Total Pages 2

First Named Inventor or Application Identifier Bart Reynolds

Express Mail Label No. EL371008575US

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, D. C. 20231

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. X Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)
2. X Specification (Total Pages 23)
(preferred arrangement set forth below)
 - Descriptive Title of the Invention
 - Cross References to Related Applications
 - Statement Regarding Fed sponsored R & D
 - Reference to Microfiche Appendix
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings (if filed)
 - Detailed Description
 - Claims
 - Abstract of the Disclosure
3. X Drawings(s) (35 USC 113) (Total Sheets 7)
4. Oath or Declaration (Total Pages)
 - a. Newly Executed (Original or Copy)
 - b. Copy from a Prior Application (37 CFR 1.63(d))
(for Continuation/Divisional with Box 17 completed) (**Note Box 5 below**)
 - i. DELETIONS OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).
5. Incorporation By Reference (useable if Box 4b is checked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. Microfiche Computer Program (Appendix)

09/649437-08/25/00

ACCOMPANYING APPLICATION PARTS

17. If a **CONTINUING APPLICATION**, check appropriate box and supply the requisite information:

of prior application No:

X	Correspondence Address Below
----------	------------------------------

Country U.S.A. TELEPHONE (408) 720-8598 FAX (408) 720-9397

Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

FEE TRANSMITTAL FOR FY 2000**TOTAL AMOUNT OF PAYMENT (\$)** \$930.00**Complete if Known:****Application No.** Not Assigned**Filing Date** August 23, 2000 (Herewith)**First Named Inventor** Bart Reynolds**Group Art Unit** Not Assigned**Examiner Name** Not Assigned**Attorney Docket No.** 003242.P017**METHOD OF PAYMENT (check one)**

1. ☒ **The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:**

Deposit Account Number 02-2666**Deposit Account Name** _____

- ☒ **Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17**

2. ☒ **Payment Enclosed:**

☒ **Check**☐ **Money Order**☐ **Other****FEE CALCULATION****1. BASIC FILING FEE**

<u>Large Entity</u>		<u>Small Entity</u>		<u>Fee Description</u>	<u>Fee Paid</u>
<u>Code</u>	<u>Fee (\$)</u>	<u>Code</u>	<u>Fee (\$)</u>		
101	690	201	345	Utility application filing fee	<u>690.00</u>
106	310	206	155	Design application filing fee	_____
107	480	207	240	Plant filing fee	_____
108	690	208	345	Reissue filing fee	_____
114	150	214	75	Provisional application filing fee	_____
SUBTOTAL (1)					\$ 690.00

2. EXTRA CLAIM FEES

			<u>Extra Claims</u>	<u>Fee from below</u>	<u>Fee Paid</u>
Total Claims	<u>29</u>	- 20** =	<u>9</u>	X \$18.00	= \$162.00
Independent Claims	<u>4</u>	- 3** =	<u>1</u>	X \$78.00	= \$78.00
Multiple Dependent					=

****Or number previously paid, if greater; For Reissues, see below.**

<u>Large Entity</u>		<u>Small Entity</u>		<u>Fee Description</u>	
<u>Code</u>	<u>Fee (\$)</u>	<u>Code</u>	<u>Fee (\$)</u>		
103	18	203	9	Claims in excess of 20	
102	78	202	39	Independent claims in excess of 3	
104	260	204	130	Multiple dependent claim, if not paid	
109	78	209	39	**Reissue independent claims over original patent	
110	18	210	9	**Reissue claims in excess of 20 and over original patent	
SUBTOTAL (2)					\$ 240.00

01/10/2000

- 1 -

PTO/SB/17 (6/99)

Patent fees are subject to annual revisions. Small Entity payments must be supported by a small entity statement, otherwise large entity fees must be paid.

See Forms PTO/SB/09-12

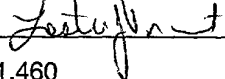
003242.P017 08454960

FEE CALCULATION (continued)**3. ADDITIONAL FEES**

<u>Large Entity</u>		<u>Small Entity</u>		<u>Fee Description</u>	<u>Fee Paid</u>
<u>Fee Code</u>	<u>Fee (\$)</u>	<u>Fee Code</u>	<u>Fee (\$)</u>		
105	130	205	65	Surcharge - late filing fee or oath	_____
127	50	227	25	Surcharge - late provisional filing fee or cover sheet	_____
139	130	139	130	Non-English specification	_____
147	2,520	147	2,520	For filing a request for reexamination	_____
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	_____
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	_____
115	110	215	55	Extension for response within first month	_____
116	380	216	190	Extension for response within second month	_____
117	870	217	435	Extension for response within third month	_____
118	1,360	218	680	Extension for response within fourth month	_____
128	1,850	228	925	Extension for response within fifth month	_____
119	300	219	150	Notice of Appeal	_____
120	300	220	150	Filing a brief in support of an appeal	_____
121	260	221	130	Request for oral hearing	_____
138	1,510	138	1,510	Petition to institute a public use proceeding	_____
140	110	240	55	Petition to revive unavoidably abandoned application	_____
141	1,210	241	605	Petition to revive unintentionally abandoned application	_____
142	1,210	242	605	Utility issue fee (or reissue)	_____
143	430	243	215	Design issue fee	_____
144	580	244	290	Plant issue fee	_____
122	130	122	130	Petitions to the Commissioner	_____
123	50	123	50	Petitions related to provisional applications	_____
126	240	126	240	Submission of Information Disclosure Stmt	_____
581	40	581	40	Recording each patent assignment per property (times number of properties)	_____
146	690	246	345	For filing a submission after final rejection (see 37 CFR 1.129(a))	_____
149	690	249	345	For each additional invention to be examined (see 37 CFR 1.129(a))	_____
Other fee (specify) _____					_____
Other fee (specify) _____					_____

SUBTOTAL (3) \$0

*Reduced by Basic Filing Fee Paid

SUBMITTED BY:Typed or Printed Name: Lester J. VincentSignature  Date August 23, 2000Reg. Number 31,460 Deposit Account User ID _____
(complete if applicable)

jc586 U.S. PTO
09/649437
08/25/00

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: EL371008575US

Date of Deposit: August 23, 2000

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Geneva Walls

(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee)

August 23, 2000

(Date signed)

Serial/Patent No.: *** Filing/Issue Date: August 23, 2000

Client: Triscend Corporation (Herewith)

Title: METHOD AND APPARATUS FOR SPECIFYING ADDRESSABILITY AND BUS CONNECTIONS IN A LOGIC DESIGN

BSTZ File No.: 003424.P017 Atty/Secty Initials: LJV/DNT/gw

Date Mailed: August 23, 2000 Docket Due Date: _____

The following has been received in the U.S. Patent & Trademark Office on the date stamped hereon:

<input type="checkbox"/> Amendment/Response (____ pgs.)	<input checked="" type="checkbox"/> Express Mail No <u>EL371008575US</u> <input checked="" type="checkbox"/> Check No <u>37220</u>
<input type="checkbox"/> Appeal Brief (____ pgs.) (in triplicate)	<input type="checkbox"/> _____ Month(s) Extension of Time Amt. <u>\$930.00</u>
<input checked="" type="checkbox"/> Application - Utility (<u>23</u> pgs., with cover and abstract)	<input type="checkbox"/> Information Disclosure Statement & PTO 149 (____ pgs.) <input type="checkbox"/> Check No _____
<input type="checkbox"/> Application - Rule 1.53(b) Continuation (____ pgs.)	<input type="checkbox"/> Issue Fee Transmittal Amt: _____
<input type="checkbox"/> Application - Rule 1.53(b) Divisional (____ pgs.)	<input type="checkbox"/> Notice of Appeal
<input type="checkbox"/> Application - Rule 1.53(b) CIP (____ pgs.)	<input type="checkbox"/> Petition for Extension of Time
<input type="checkbox"/> Application - Rule 1.53(d) CPA Transmittal (____ pgs.)	<input type="checkbox"/> Petition for _____
<input type="checkbox"/> Application - Design (____ pgs.)	<input checked="" type="checkbox"/> Postcard
<input type="checkbox"/> Application - PCT (____ pgs.)	<input type="checkbox"/> Power of Attorney (____ pgs.)
<input type="checkbox"/> Application - Provisional (____ pgs.)	<input type="checkbox"/> Preliminary Amendment (____ pgs.)
<input type="checkbox"/> Assignment and Cover Sheet	<input type="checkbox"/> Reply Brief (____ pgs.)
<input checked="" type="checkbox"/> Certificate of Mailing (Express Mail)	<input type="checkbox"/> Response to Notice of Missing Parts
<input type="checkbox"/> Declaration & POA (____ pgs.)	<input type="checkbox"/> Small Entity Declaration for Indep Inventor/Small Business
<input type="checkbox"/> Disclosure Docs & Orig & Copy of Inventor's Signed Letter (____ pgs.)	<input checked="" type="checkbox"/> Transmittal Letter, in duplicate (2 pgs.)
<input checked="" type="checkbox"/> Drawings <u>7</u> # of sheets includes <u>7</u> figures	<input checked="" type="checkbox"/> Fee Transmittal, in duplicate (2 pgs.)

☒ Other: Copy of Express Mail Stamp.

UNITED STATES PATENT APPLICATION
FOR
METHOD AND APPARATUS FOR
SPECIFYING ADDRESSABILITY AND BUS CONNECTIONS
IN A LOGIC DESIGN

First-Named Inventor:

Bart Reynolds

Prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

12400 WILSHIRE BOULEVARD

SEVENTH FLOOR

LOS ANGELES, CALIFORNIA 90025

(408) 720-8598

Attorney's Docket No. 003424.P017

"Express Mail" mailing label number EL371008575US

Date of Deposit August 23, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service
"Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above
and is addressed to the Assistant Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Geneva Walls

(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee)

METHOD AND APPARATUS FOR SPECIFYING ADDRESSABILITY AND BUS CONNECTIONS IN A LOGIC DESIGN

FIELD OF THE INVENTION

The present invention relates generally to the field of logic design. More
5 specifically, the present invention is directed to a method and an apparatus for
specifying addressability and bus connections.

BACKGROUND

Logic designers use hardware description language (HDL) or schematic
capture to model a circuit at different level of abstractions. The circuit model is
10 synthesized to construct a gate-level netlist. System designs that include
memory-mapped devices require the logic designer to fully specify the
addressability and bus connections of the memory-mapped device in the logic
design. Great caution is exercised when specifying the addressability and bus
connections for memory-mapped devices interacting with multi-byte system
15 buses. Traditional electronic design automation tool flows require the
addressability and data connections of a device to a system bus to be explicitly
specified. This includes address matching, lane matching, connections to system
bus data bits and any other auxiliary logic.

In the case of an 8-bit system bus, where only byte-wide transactions are
20 supported, the specifications of addressability and bus connections is fairly
obvious. The lane-matching function is unnecessary because all transactions are
byte-wide. All devices connect to the same bits (bits 7:0) of the data bus.
Although specifying the addressability and bus connections may be tedious
when designing for an 8-bit system bus, there is little danger of accidentally
25 specifying inconsistent addressability and bus connections.

In the case of a 32-bit system bus, where byte-wide, halfword-wide, and word-wide transactions are supported, the connections between the device and the system bus are much more complex. The interdependencies between the address-matching function, the lane-matching function, and the connections to the data bus make it much more likely that the logic designer will accidentally

Because the bus connections and lane-matching function must be consistent with the address-matching function, it is not possible to change the address of a memory-mapped device without invalidating the lane-matching function and bus connectivity.

SUMMARY OF THE INVENTION

In one embodiment, a method for specifying addressability in a memory-mapped device is disclosed. A data access primitive is used to model addressability for the memory-mapped device. Addressability comprises an address matching function, a lane matching function and one or more bus connections. A first starting address for the memory-mapped device is specified. A first set of addressing matching function, lane matching function and one or more bus connections for the memory-mapped device is generated using the data access primitive and the first starting address.

Other features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example in the following drawings in which like references indicate similar elements. The following drawings disclose various embodiments of the present invention for purposes of illustration only and are not intended to limit the scope of the invention.

Figure 1 is an exemplary logic diagram showing explicit addressability and bus connections.

Figure 2 is an exemplary diagram of a halfword selector data access primitive.

Figure 3 is an exemplary diagram of another halfword selector data access primitive.

Figure 4 is an exemplary diagram of a byte selector data access primitive.

Figure 5 is an exemplary logic diagram using a data access primitive.

Figure 6 illustrates an embodiment of a computer system that can be used with the present invention.

Figure 7 illustrates an embodiment of a computer-readable medium.

DETAILED DESCRIPTION

In one embodiment, a method for specifying addressability for a memory-mapped device is disclosed. An intended advantage of this method is to simplify the task of the logic designer when specifying the addressability and bus connections of a memory-mapped logic device.

Figure 1 is an exemplary logic diagram showing explicit address-matching functions, lane matching functions and data bus connections. **Figure 1** shows a memory mapped 2-byte device (e.g., register) with individually addressable bytes 120, 125. Other signals not shown in Figure 1 may include, for example, addresses, data, clock, wait, read, write, etc. Address-matching function 105 is typically specified as a set of logic gates (in schematic capture) or a logic equation (in hardware description language ("HDL")) that is synthesized to a set of gates. The address-matching function 105 determines the addresses the memory-mapped device is mapped to. The address-matching function 105 also performs address-decoding function. At design time, a constant address is specified. At run time, during operation of the logic, the address decoding function compares the constant against the addresses seen on the bus to see if there is a match.

The lane-matching function which includes Lane Match 0 110 and Lane Match 1 115, is also specified as a set of logic gates (in schematic capture) or a logic equation (in HDL) that is synthesized to a set of gates. The lane-matching function suppresses the address-matching function for certain bus transaction sizes and alignments. For example, for the 2-byte register with the individually addressable bytes shown in **Figure 1**, there may be a single address-matching function 105 with a distinct lane-matching function for each byte. The Lane Match 0 lane-matching function 110 would match all transactions (e.g., read, write) that include the first byte. The Lane Match 1 lane-matching function 115

would match all transactions that include the second byte. The logic diagram of **Figure 1** would require the logic designer to explicitly specify connections to system bus data bits. The first byte 120 of the 2-byte register would connect to a set of eight different system bus data bits (0 to 7) 130. Similarly, the second byte 125 of the 2-byte register would connect to another set of eight different system bus data bits (8 to 15) 135.

For example, in a 2-byte transaction to the register at address 0x00000004, when there is a match, the Lane Match 0 lane-matching function 110 would match the first byte address 0x00000004. The Lane Match 1 lane-matching function 115 would match the second byte address 0x00000005. The address 0x00000004 refers to the first byte 120 of the register, and the address 0x00000005 refers to the second byte 125 of the register. When there is a write transaction, the first data byte 130 is written into the first byte of the register 120 and the second data byte 135 is written into the second byte of the register 125. When there is a read transaction, the first data byte 140 and the second data byte 145 from the register is provided to the respective bits of the data bus.

The address 0x00000004 in this example is a constant specified at design time. When there is a need to make any changes to the design of the memory-mapped device, such as, for example, the address constant, the logic designer has to make the change to the lane matching and the related connections. For example, for a 32-bit data bus, when the logic designer wants to change the address from 0x00000004 to 0x00000006, this requires the lane matching function to match the first byte to Lane Match 2 (not shown) and the second byte to Lane Match 3 (not shown). The corresponding data bytes would be from bits 15-23 for the first data byte and bits 24-31 for the second data byte (i.e., the second half of the word). This would require having to change and recompile the HDL source and the schematic.

In one embodiment, the method of the present invention allows the logic designer to specify addresses for an addressable entity without having to be involved in the detailed requirement of address matching and lane matching. The logic designer uses a set of logic design components, referred to herein as "data access primitives", to specify an assembly of address and lane-matching logic and associated data bus connections. The data access primitive hides the details of interconnection to the bus, and abstracts away the interdependency of address-matching functions, lane-matching functions, and data bus connections.

Figure 2 is an exemplary diagram of a halfword selector data access primitive. Each data access primitive implies an address-matching function, one or more lane-matching functions, and bus connections for one or more bytes of data, as well as auxiliary logic. The data access primitive in **Figure 2** is referred to herein as "HALFSEL" data access primitive. The HALFSEL data access primitive is a fully addressable data access primitive because it can be used to connect a byte-, a halfword-, or a word-addressable 2-byte entity to the data bus.

The write-select (WRSEL) port 205 has two lines, one for each byte of the halfword. During a halfword or word write transaction, both lines of the WRSEL port 205 go high when there is an address match. During a byte write transaction, at most one of the lines of the WRSEL port 205 goes high when there is an address match. Similarly, the read-select (RDSEL) port 208 has two lines and goes active during a read transaction when the addresses and lanes match. The HALFSEL data access primitive includes a data write port (DW) 210 and a data read port (DR) 215. The data read port provides data from the device to the bus. The data write port receives data from the bus. For read-only data, the WRSEL port 205 and the DW port 210 are not connected. For write-only data, the DR port 215 is tied low.

The physical port 220 has the address constant indicating the starting address of the memory-mapped device. The autowait (AWAIT) port 225 is a constant flag. When the AWAIT flag is high, one wait state is automatically generated to indicate to a device reading this address that it is going to take an additional bus clock cycle to get the data out of the memory mapped device. When the flag is low, there is no wait state. In another embodiment, the AWAIT port 225 can be configured to be multiple bits wide to enable encoding of additional wait states. One skilled in the art would recognize that other wait states beyond those asserted by the data access primitive may be asserted by other logic in the design, depending on the needs of the design.

Figure 3 is an exemplary diagram of another halfword selector data access primitive. The data access primitive in **Figure 3** is referred to herein as "HALFSELH" data access primitive. The HALFSELH data access primitive is a restricted data access primitive because it can be used to connect a halfword-, or a word-addressable half-word entity to the data bus. The HALFSELH can not be used to address a byte of the half-word entity, which is different from the fully addressable HALFSEL data access primitive. The write-select (WRSEL) port 305 has one line which is shared by both bytes of the halfword. During a halfword or word write transaction, the line goes high when there is an address match. The read-select (RDSEL) port 308 has one line and goes high during a read transaction when the address matches. The HALFSELH data access primitive includes a data write port (DW) 310 and a data read port (DR) 315.

Figure 4 is an exemplary diagram of byte selector data access primitive. The data access primitive in **Figure 4** is referred to herein as "BYTESEL" data access primitive. The BYTESEL data access primitive is a fully addressable data access primitive because it can be used to connect a one-byte entity to the data bus. The BYTESEL data access primitive is very similar to the HALFSEL data

access primitive, except that the write-select (WRSEL) port 405 and the read-select (RDSEL) 408 each has one line (bit) instead of two lines. The BYTESEL data access primitive also differs from the HALFSEL data access primitive in that it matches only a single byte rather than two bytes.

5 The data access primitives do not instantiate the logic (e.g., registers or RAMs) that stores the data being accessed. The data access primitives provide the addressability and data bus connections for the logic. Using the data access primitive such as, for example, the HALFSEL or the HALFSELH, the logic designer does not have to be involved with the complication of lane matching or
10 addressability at design time or at subsequent changes. One skilled in the art will recognize that other data access primitives such as, for example, WORDSEL and WORDSELW, can also be implemented using the descriptions described above.

 Although a data access primitive may require the logic designer to
15 specify an explicit starting address for the physical port, the logic designer may leave the starting address of a data access primitive unspecified. The logic designer may choose to allow the starting address to be automatically assigned by an address allocator. The logic designer may choose to specify or restrict the starting addresses to be assigned to data access primitives using one or more
20 address constraints. For example, the address constraints may be a block of addresses to be excluded or a specific starting address. In one embodiment, the address allocator is a software program that ensures that all of the data access primitives have fully specified addressability information. The address allocator reconciles the addressability specified in the logic design with the address
25 constraints specified by the logic designer.

 In one embodiment, a mapper program, referred to herein as a data access technology mapper, converts the data access primitives into low-level

logic components necessary to implement the address-matching function, lane-matching function, bus connections, and auxiliary logic described above in **Figure 1**. The data access technology mapper replaces the data access primitives with low-level logic components whose type and interconnection depend on both the type of the data access primitive and the starting address. The starting address may be either allocated by the address allocator or specified by the user. The data access technology mapper uses the starting address which is allocated by the address allocator and decides how the lane matching should be done among the components and which data should be read from the register. The exact mapping from data access primitives to low-level logic components depends on the implementation technology targeted by the data access technology mapper. For example, in the case of a Configurable System-on-Chip (CSoC), the data access technology mapper converts each data access primitives into one or more address selectors and multiple socket primitives for connecting to the system bus signals.

In one embodiment, the data access technology mapper combines the addressability implied by the data access primitives and the output of the address allocator program. By incorporating a data access primitive into a design, the logic designer can specify a complex assembly of address and lane-matching logic and associated data bus connections easily and without risk of specifying inconsistent information. At a later time, the logic designer can change the address for the data access primitive just by changing the address constraints. The logic designer does not have to change the logic design.

The data access primitives in the present invention are not implemented as a traditional logic macro. Although the data access primitive simplifies and abstracts the specification of logic design, the data access primitive is not a fixed composition of lower-level logic components. The data access technology

mapper decides how to decompose the data access primitive, and that decomposition is dependent directly on the address assigned to the data access primitive. For example, depending on the address specified by the logic designer, the HALFSEL data access primitive may be converted by the data access technology mapper into different implementations.

Figure 5 is an exemplary logic diagram showing a HALFSEL data access primitive in an equivalent logic to that in **Figure 1**. The HALFSEL data access primitive 505 includes the data read (DR) port, the data write (DW) port and the data write select (DWSEL) port. The DWSEL port has two lines. Although not shown, the HALFSEL 505 also include the ports shown in **Figure 2**. The HALFSEL 505 is connected to a two-byte register with individually addressable bytes 510 and 520. Each of the two write-select lines of the HALFSEL data access primitive 505 is connected to the write-enable input of one of the 8-bit registers 510 and 520. Each of the two bytes of DW port of the HALFSEL data access primitive 505 is connected to the data input of one of the 8-bit registers 510 and 520. The data output of the register 510 and the register 520 are combined to form the two bytes of data input to the HALFSEL data access primitive 505 and is connected to the DR port.

Assume that the address allocator program assigns the address 0x00000004 to the HALFSEL data access primitive 505. The data access technology mapper program converts the HALFSEL data access primitive 505 into a single address-matching function and two lane-matching functions. The address-matching function matches either address 0x00000004 or 0x00000005 (two bytes in the halfword). The first lane matching function matches only transactions that include the byte at 0x00000004. The second lane matching function matches only transactions that include the byte at 0x00000005. The data access technology mapper program also produces sixteen connections to the

data-write (DW) port and sixteen connections to the data-read (DR) port. It also produces other auxiliary logic and connections. For example, in a 32-bit system bus, there are four transactions that pass the address matching function (i.e., matches):

- 5 1. A word-wide transaction at 0x00000004
2. A halfword-wide transaction at 0x0000004
3. A byte-wide transaction at 0x00000004
4. A byte-wide transaction at 0x00000005.

The transactions 1, 2, and 3 match the first lane-matching function (transactions
10 that contain 0x00000004). The transactions 1, 2, and 4 match the second lane-matching function (transactions that contain 0x00000005).

Figure 6 illustrates an embodiment of a computer system that can be used with the present invention. The various components shown in **Figure 6** are provided by way of example. Certain components of the computer in **Figure 6**
15 can be deleted from the addressing system for a particular implementation of the invention. The computer shown in **Figure 6** may be any type of computer including a general-purpose computer.

Figure 6 illustrates a system bus 600 to which various components are coupled. A processor 602 performs the processing tasks required by the
20 computer. Processor 602 may be any type of processing device capable of implementing the method discussed above. An input/output (I/O) device 604 is coupled to bus 600 and provides a mechanism for communicating with other devices coupled to the computer. A read-only memory (ROM) 606 and a random access memory (RAM) 608 are coupled to bus 600 and provide a storage
25 mechanism for various data and information used by the computer. Although ROM 606 and RAM 608 are shown coupled to bus 600, in alternate

embodiments, ROM 606 and RAM 608 are coupled directly to processor 602 or coupled to a dedicated memory bus (not shown).

A video display 610 is coupled to bus 600 and displays various information and data to the user of the computer. A disk drive 612 is coupled to bus 600 and provides for the long-term mass storage of information. Disk drive 612 may be used to store various software programs including the data access technology mapper program and the address allocator program. Disk drive 612 may also store the data access primitives and the source HDL programs used by the logic designer to model the circuit. Disk drive 612 may also store a synthesis program. A keyboard 614 and pointing device 616 are also coupled to bus 600 and provide mechanisms for entering information and commands to the computer. A printer 618 is coupled to bus 600 and is capable of creating a hard-copy of information generated by or used by the computer.

Figure 7 illustrates an embodiment of a computer-readable medium 700 containing various sets of instructions, code sequences, configuration information, and other data used by a computer or other processing device. The various information stored on medium 700 is used to perform various data processing operations. Computer-readable medium 700 is also referred to as a processor-readable medium. Computer-readable medium 700 can be any type of magnetic, optical, or electrical storage medium including a diskette, magnetic tape, CD-ROM, memory device, or other storage medium.

Computer-readable medium 700 includes interface code 705 that controls the flow of information between various devices or components in the computer system. Interface code 705 may control the transfer of information within a device (e.g., between the processor and a memory device), or between an input/output port and a storage device. Additionally, interface code 705 may control the transfer of information from one device to another. Computer-

readable medium 700 may also includes the data access technology mapper program 710, the address allocator program 715, and the data access primitives 720.

Thus, using the method disclosed, the logic designer can leave the address of a memory-mapped device unspecified, allowing the address allocator and data access technology mapper to decide the details of address-matching, lane-matching, and bus connectivity. The logic designer can change the addresses assigned to memory-mapped logic devices without changing the logic design.

From the above description and drawings, it will be understood by those of ordinary skill in the art that the particular embodiments shown and described are for purposes of illustration only and are not intended to limit the scope of the invention. Those of ordinary skill in the art will recognize that the invention may be embodied in other specific forms without departing from its spirit or essential characteristics. References to details of particular embodiments are not intended to limit the scope of the claims.

CLAIMS

What is claimed is:

- 1 1. A method of addressing a memory-mapped device, comprising:
2 using a data access primitive to model addressability for the memory-
3 mapped device, addressability comprising an address matching
4 function, a lane matching function and one or more bus connections,
5 specifying a first starting address for the memory-mapped device; and
6 generating a first set of addressing matching function, lane matching
7 function and one or more bus connections for the memory-mapped
8 device using the data access primitive and the first starting address.
- 1 2. The method of claim 1, further comprising generating a second set of
2 addressing matching function, lane matching function and one or more bus
3 connections for the memory-mapped device using the data access primitive
4 and a second starting address.
- 1 3. The method of claim 1, further comprising:
2 coupling the data access primitive to the memory-mapped device; and
3 coupling an address bus to the data access primitive.
- 1 4. The method of claim 3, wherein the addressing matching function
2 compares an address from the address bus with the first starting address

10 device using the data access primitive and the first starting address.

1 11. The computer readable medium of claim 10, further comprising generating
2 a second set of addressing matching function, lane matching function and
3 one or more bus connections for the memory-mapped device using the
4 data access primitive and a second starting address.

1 12. The computer readable medium of claim 10, further comprising:
2 coupling the data access primitive to the memory-mapped device; and
3 coupling an address bus to the data access primitive.

1 13. The computer readable medium of claim 12, wherein the addressing
2 matching function compares an address from the address bus with the first
3 starting address for the memory-mapped device.

1 14. The computer readable medium of claim 13, wherein the first starting
2 address is specified by a user.

1 15. The computer readable medium of claim 13, wherein the first starting
2 address is generated automatically.

1 16. The computer readable medium of claim 15, wherein the first starting
2 address is generated automatically using a set of address constraints.

1 17. The computer readable medium of claim 10, wherein the memory-mapped
2 device is selected to allow addressability for a minimum size transaction
3 supported by the memory-mapped device.

1 18. A method, comprising:

2 selecting a data access primitive that provide data access of a desired
3 transaction size, the data access primitive implying an addressing
4 matching function, a lane matching function and one or more bus
5 connections for a memory-mapped device;

6 specifying an address constraint for the memory-mapped device;

7 instantiating a logic for the memory-mapped device, comprising:

8 generating a starting address for the memory mapped device using
9 the address constraint;

10 using the selected data access primitive and the starting address to
11 map the logic for the memory mapped device capable of being
12 accessed at the desired transaction size, comprising:

13 generating the address matching function, and

14 generating the lane matching function and the one or more bus
15 connections.

1 19. The method of claim 18, wherein the address constraint is specified by a
2 user, and wherein the starting address for the memory mapped device is
3 generated automatically.

- 1 20. The method of claim 18, wherein the transaction size is one in a group
2 comprising a byte, a halfword and a word.
- 1 21. The method of claim 18, further comprising using a new starting address
2 for the memory-mapped device without having to specify changes to the
3 addressing function, the lane matching function and the one or more bus
4 connections.
- 1 22. The method of claim 21, wherein a different logic for the memory mapped
2 device is instantiated automatically using the same data access primitive
3 and the new starting address.
- 1 23. The method of claim 18, wherein the addressing matching function
2 compares an address from an address bus coupled with the data access
3 primitive with the starting address, and wherein when there is match, the
4 lane matching function matching the transaction size of a transaction to a
5 respective section of the memory-mapped device.
- 1 24. A computer readable medium containing executable instructions which,
2 when executed in a processing system, causes the processing system to
3 perform the steps of a method, comprising:
4 selecting a data access primitive that provide data access of a desired
5 transaction size, the data access primitive implying an addressing
6 matching function, a lane matching function and one or more bus
7 connections for a memory-mapped device;

8 specifying an address constraint for the memory-mapped device;

9 instantiating a logic for the memory-mapped device, comprising:

10 generating a starting address for the memory mapped device using
11 the address constraint;

12 using the selected data access primitive and the starting address to
13 map the logic for the memory mapped device capable of being
14 accessed at the desired transaction size, comprising:

15 generating the address matching function, and

16 generating the lane matching function and the one or more bus
17 connections.

1 25. The computer readable medium of claim 24, wherein the address constraint
2 is specified by a user, and wherein the starting address for the memory
3 mapped device is generated automatically.

1 26. The computer readable medium of claim 24, wherein the transaction size is
2 one in a group comprising a byte, a halfword and a word.

1 27. The computer readable medium of claim 24, further comprising using a
2 new starting address for the memory-mapped device without having to
3 specify changes to the addressing function, the lane matching function and
4 the one or more bus connections.

1 28. The computer readable medium of claim 27, wherein a different logic for
2 the memory mapped device is instantiated automatically using the same
3 data access primitive and the new starting address.

1 29. The computer readable medium of claim 24, wherein the addressing
2 matching function compares an address from an address bus coupled with
3 the data access primitive with the starting address, and wherein when
4 there is match, the lane matching function matching the transaction size of
5 a transaction to a respective section of the memory-mapped device.

ABSTRACT OF THE DISCLOSURE

In one embodiment, a method for specifying addressability in a memory-mapped device is disclosed. A data access primitive is used to model
5 addressability for the memory-mapped device. Addressability comprises an address matching function, a lane matching function and one or more bus connections. A first starting address for the memory-

00000000000000000000000000000000

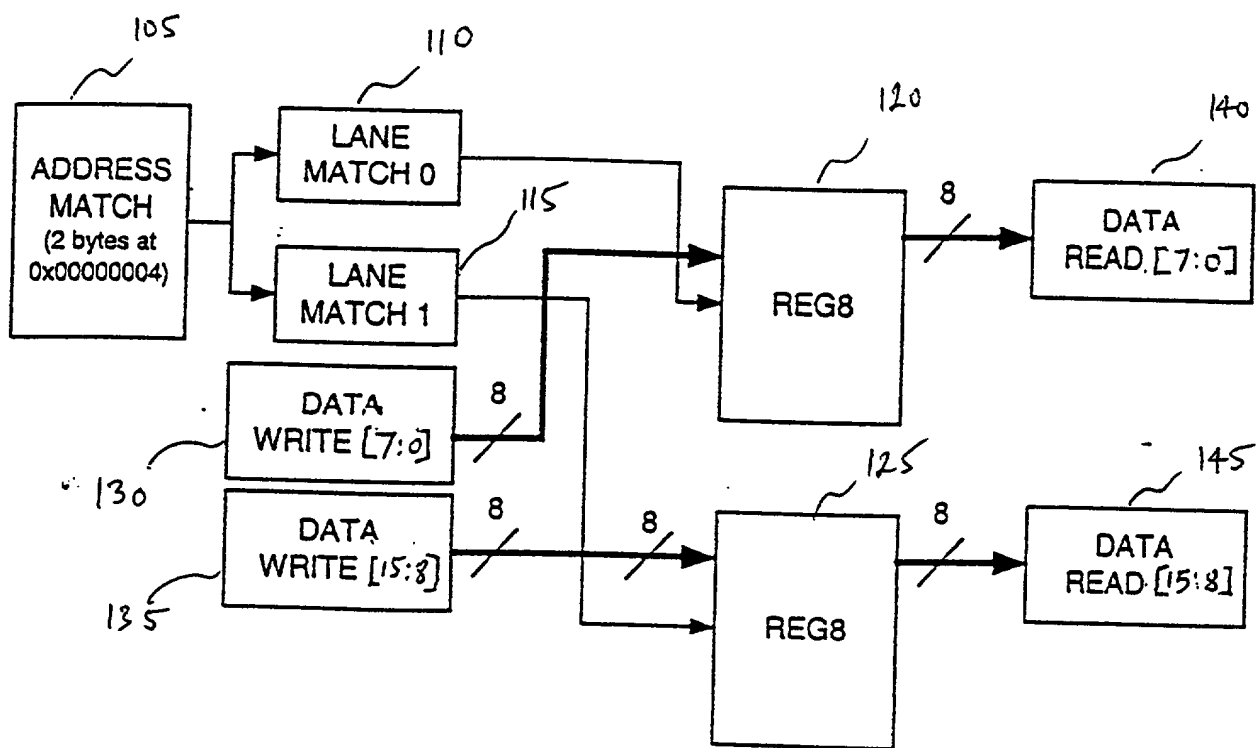


FIGURE 1

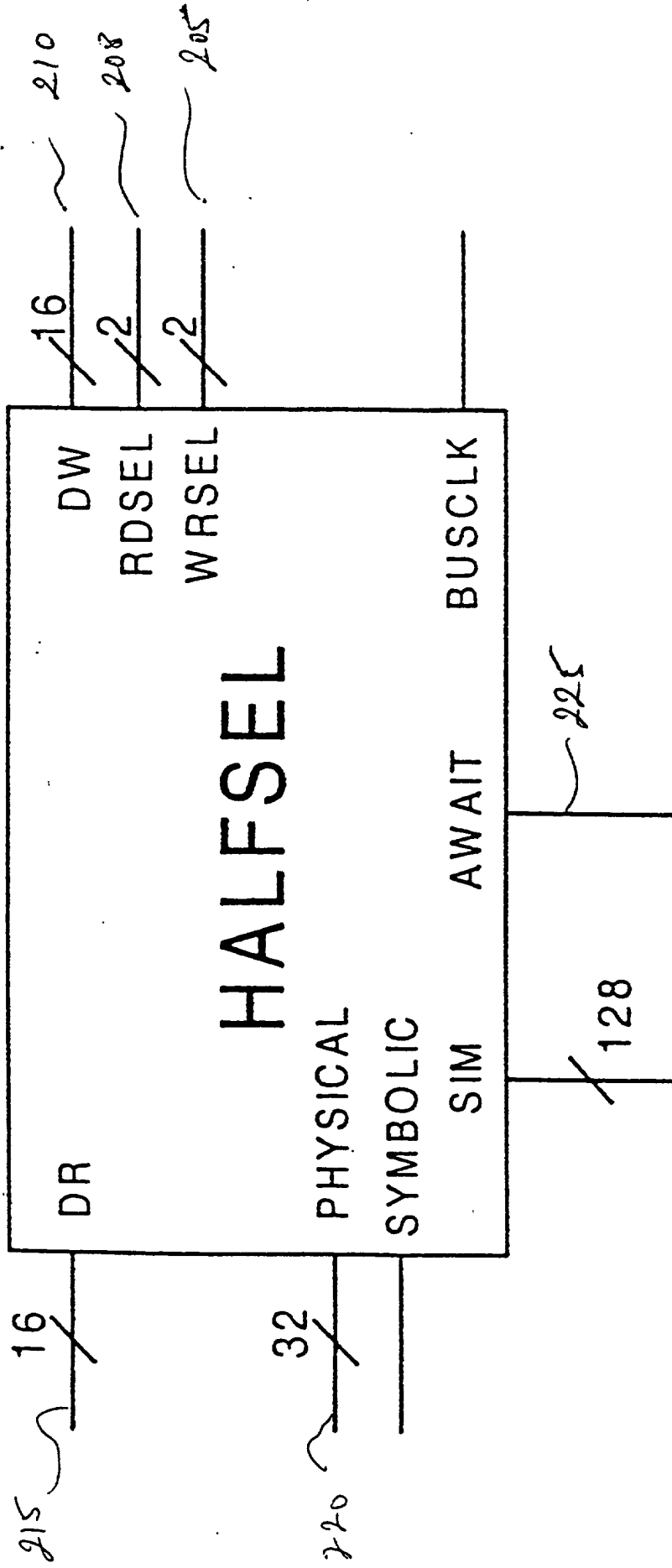


FIGURE 2

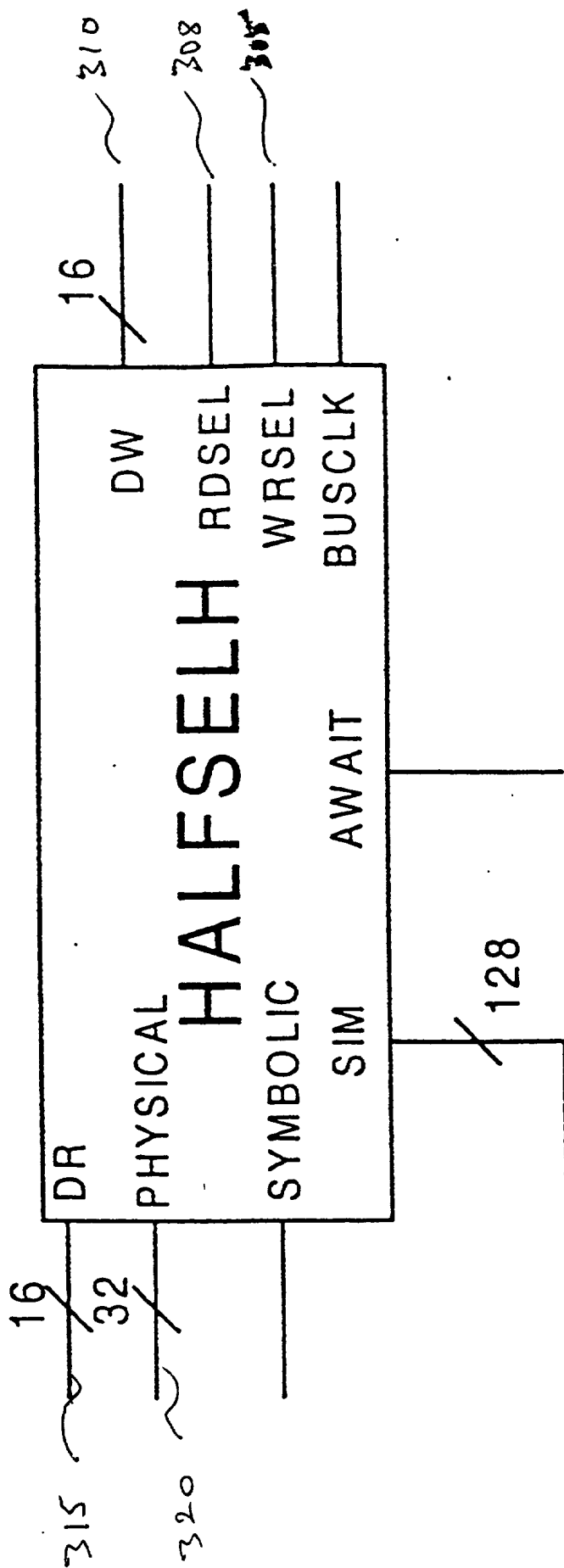


FIGURE 3

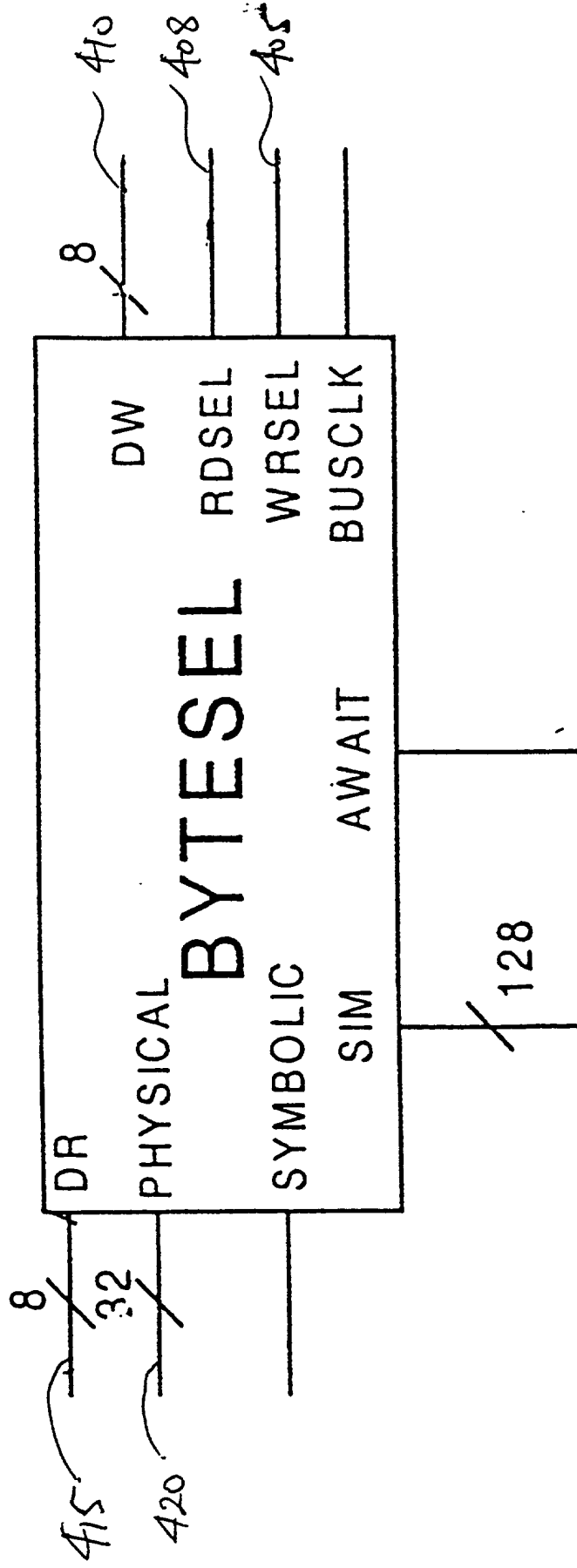


FIGURE 4

The diagram illustrates a data path architecture. A 16-bit input signal enters a block labeled 'HALFSEL' (505). This block has two outputs: a 16-bit signal 'DW' and a 2-bit signal 'WRSEL'. The 16-bit 'DW' signal is split into two 8-bit paths. Each 8-bit path passes through an 8-bit register, labeled 'REG8' (510 and 520). The outputs of these registers are then combined back into a single 16-bit output signal.

FIGURE 5

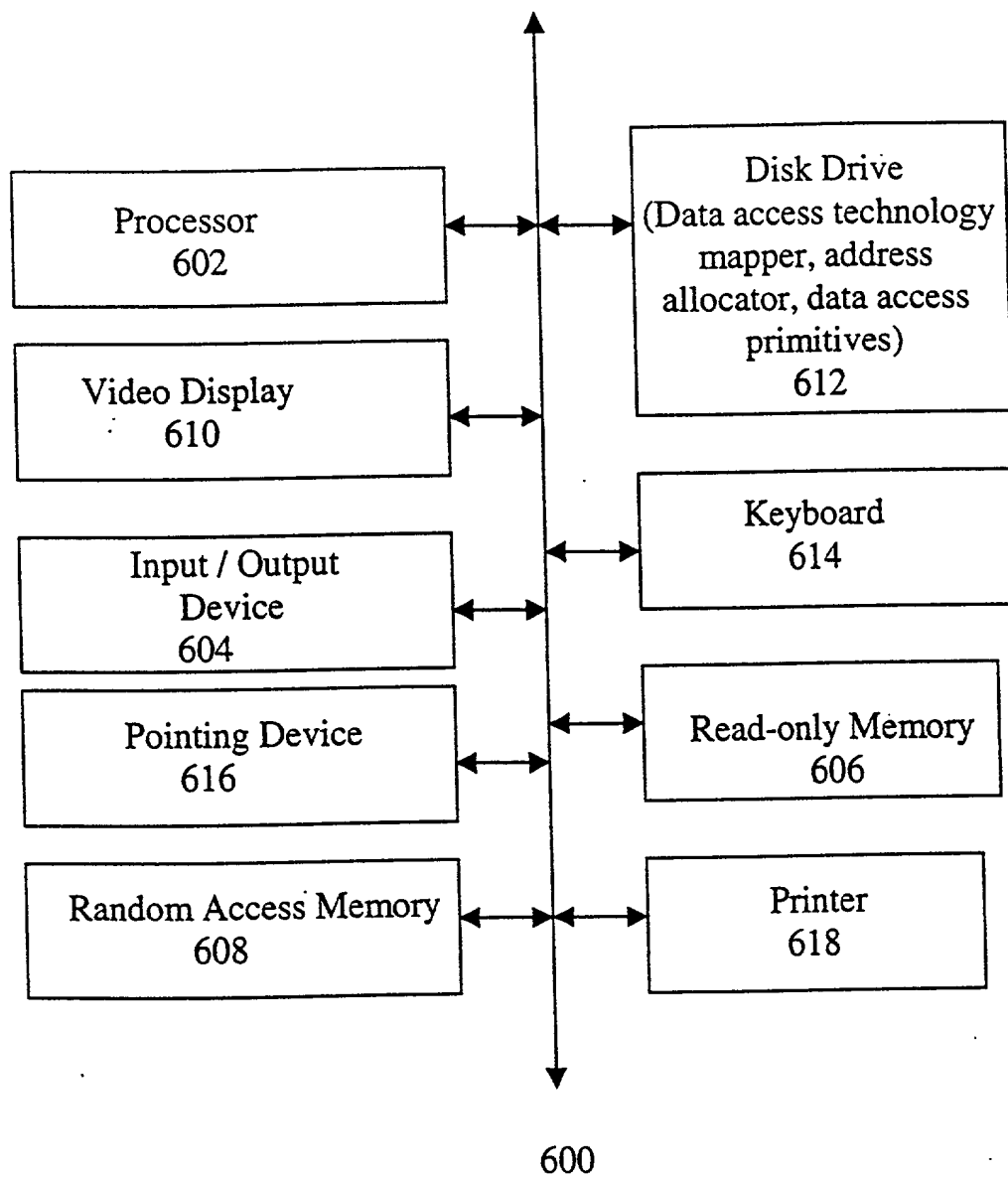


FIGURE 6

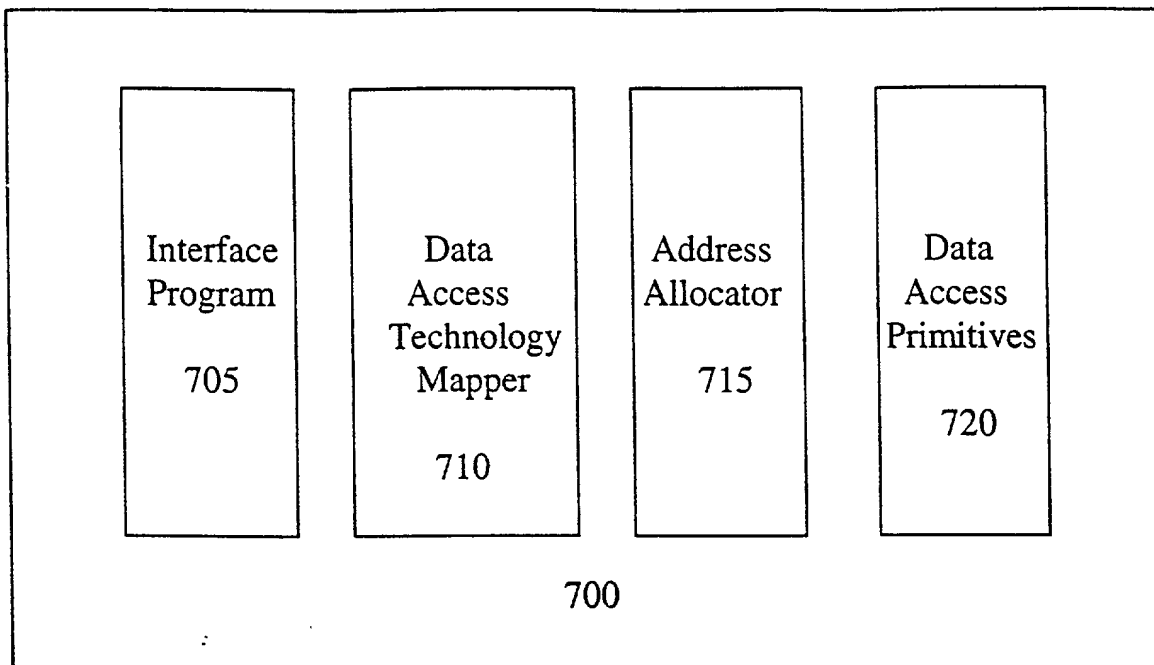


Figure 7